Parser Hybridisation for Natural Languages

The University of Manchester

Manchester, United Kingdon

Abstract

Identifying and establishing structural relations between words in natural language sentences is called *Parsing*. Ambiguities in natural languages make parsing a difficult task. Parsing is more difficult when dealing with a structurally complex natural language such as Arabic, which contains a number of properties that make it particularly difficult to handle. In this paper, we briefly highlight some of the complex structure of Arabic, and we identify different parsing approaches (grammar-driven and data-driven approaches) and briefly discuss their limitations. Our main goal is to combine different parsing approaches and produce a hybrid parser, which retains the advantages of data-driven approaches but is guided by grammatical rules to produce more accurate results. We describe a novel technique for directly combining different parsing approaches. Results for initial experiments that we have conducted in this work, and our plans for future work is also presented.

Keywords: Parsing, Hybrid Parsing, Natural Language Processing, Dependency Parsing

1. Introduction

Establishing the structural relations between natural language words is called *parsing* (Aho and Ullman, 1972, p.63). Parsing is one of the core components of many natural language processing applications (Farghaly and Shaalan, 2009), such as: Machine Translation, Speech recognition, and dialogue based Systems. However, parsing is a challenging task due to language ambiguities (Collins, 2003), which is caused by multiple interpretations of words, word order freedom, and missing items. Hence, adequate parsers are often unavailable, particularly for languages with complex structures such as Arabic (Lee et al., 2008, p.82).

It is desirable that parsers have three main features: (i) efficiency (consuming as little time as possible), (ii) robustness (successfully parsing a large proportion of input strings), and (iii) accuracy (produce correct results). It has been argued that it is not possible to achieve all three features at once (Nivre, 2006). Our goal is to optimise speed and accuracy while maintaining a reasonable level of robustness by combining features of data-driven and grammar-driven approaches. We test our parser on Arabic because Arabic is structurally complex, which makes it hard to parse and it will act as a rigorous test-bed for our approach.

2. Natural language parsing approaches

There are wo approaches to parsing natural languages: (i) grammar-driven, and (ii) data-drives. Each approach has some limitations. A parser is considered robust if it can analyse a large proportion of a set of natural language sentences. Grammar-driven approaches normally lack robustness because sometimes grammatical rules of the natural language may not exist in order for parsers to be able to produce analyses for a given input string, and because a given input string may be understandable by a speaker but it is not part of the natural language, e.g., when a word in misspelled, or if material is omitted in a sentence. Some researchers argue that robustness problem in grammar-driven approaches can be solved by relaxing grammar constraints in parsers (Samuelsson and Wir'n, 2000). However, relaxing grammars may result in parsers producing several analyses for a given input string, which means that parsers will consume more time and computing resources for exploring these analyses which may consequently affect efficiency. Moreover, having several candidate analyses for a given input string may increase the risk of parsers selecting an incorrect analysis as the final result, which could aggravate the problem of accuracy.

Data-driven parsers on the other hand, use an inductive mechanism for mapping input strings to output analyses. According to Nivre (2006), in most existing data-driven parsers any input strings are assigned at least one analysis, which means that data-driven parsers are highly robust. However, the extreme robustness of data-driven parsers means that they will assign analyses that are probably grammatically incorrect, hence data-driven parsers may not produce highly accurate parse results compared with its counterpart grammar-driven parsers.

Furthermore, the problem of disambiguation can be severe in data-driven parsers because the improved robustness is the result of extreme constraints relaxation. But, this is compensated by the fact that the inductive inference scheme, which is obtained by using machine learning algorithms in data-driven parsers, provides a mechanism for disambiguation by associating a score with each analysis intended to reflect some optimality criterion, or by implicitly maximising this criterion in a deterministic selection. Regarding the problem of efficiency, it is argued that data-driven approaches is superior to grammar-driven approaches (Nivre, 2006), but it is often at the expense of less accurate output (Kaplan et al., 2004)

3. Arabic

Ambiguities in natural languages affect parsers' efficiency, robustness and accuracy, hence it is a major problem in parsing natural language sentences (Baptista,

1995; Collins, 2003). Arabic has a complex syntactic structure (Daimi, 2001), which makes it particularly challenging to parse, below are some of the major source of ambiguities in Arabic.

- **Word order freedom:** The canonical order of Arabic sentences is VSO. But, Arabic has a high degree of syntactic flexibility (Daimi, 2001), hence a range of other word orders such as VOS, SVO and OVS are also possible (Ramsay and Mansour, 2006), which is considered as a source of ambiguities in Arabic (Attia, 2008, p.179) because reordering words in Arabic sentences makes it hard to distinguish between nominative and accusative cases. For example, in the sentence "Ahmad yahatarm A*li*" "*Ahmed respects Ali*" it is clear that "*Ahmed*" is the subject in the sentence and "*Ali*" is the object. But, reordering the words as "*yahtarm 'Ahmad Ali*" "*respects Ahmed Ali*" means that the subject could be either "*Ahmed*" or "*Ali*" which lead to structural ambiguity.
- **Clitics:** Clitics are morphemes that possess the syntactic characteristics of a word but they are morphologically bound to other words (Crystal, 1980). Arabic clitics could be attached to the start or end of words, which it often alters their formation, such as from nouns to verbs, or even alters the verb type from transitive to intransitive (Nelken and Shieber, 2005).

Arabic conjunctions often appear as clitics and they modify Arabic verbs. For instance, the sentence "*wlyahum AlyuN fy AlmasAla*" "*Ali is the leader in their situation*" where "*wlyahum*" "*their leader*", which is a noun, is ambiguous because the letters "*w*" and "*l*" could be clitics attached to the word "*yahum*" "*take charge*" and can modify these words into verbs, as in the sentence "*wlyahum AlyuN fy AlmasAla*" "*and Ali to take charge of the situation*", where the word is a verb.

- **Noun multifunctional:** It is difficult to define Arabic nouns in comparison to its verbs because they encompass a wide range of categories. One of the reasons that Arabic nouns create ambiguities is that some nouns are derived from verbs, and they can function as verbs sometimes (Attia, 2008). e.g., "*Albahth*" "*search*" can function as a noun as in "*istamarra Altilmythu fy Albahthy liljAmiaT*" "*the student continued in his research for the university*", and as a verb as in "*istamarra Altilmythu fy Albahthy an AljAmiaT*" "*the student continued searching for the university*"
- **Pro-drop:** The subject of a sentence could be omitted if the verb's agreement features are rich enough to recover its content (Chomsky, 1981). Arabic verbs recover missing subjects by conjugating themselves to indicate the gender, number and person of the omitted pronoun subject (Attia, 2008). Arabic pronouns may be omitted if the verb can recover them, as in, "*Akalat Al dajAjT*" "*ate the chicken*". The verb "*Akalat*" "*ate*" indicates that the missing subject is a singular, feminine, and third person pronoun. In Arabic, verbs can be transitive and also intransitive when a pronoun is dropped. It is not clear from the above sentence that the noun phrase (NP) "*Al dajAjT*" "*the chicken*" following the verb "*Akalat*" "*ate*" is the subject. The sentence would mean the chicken was eaten and the verb "*Akalat*" "*ate*" is intransitive if the NP is the subject. But, the sentence would mean "*she ate the chicken*" and the verb "*Akalat*" "*ate*" is transitive if the NP is the object of the verb and the subject is an omitted pronoun (as "*she*"). Hence, due to pro-drops, parsers generate different structural analysis.

4. Related work

Combining data-driven and grammar-driven approaches is increasingly becoming popular. Some works involved combining state-of-art dependency data-driven parsers, such as MaltParser (Nivre, 2006) and MSTParser (MacDonald, 2006), while some other works focused on combining data-driven and grammar-driven approaches. The latter is the type of work that is more relevant to the work we present in this paper, However, they are not directly combining them, they instead use the output of one parser as input to another parser.

Øvrelid et al(2009) combined a grammar-driven parser, that was based on Lexical Functional Grammar (LFG), with a data-driven parser (MaltParser). In their approach, they supply MaltParser with outputs from the LFG parser. The LFG parser outputs phrase structured trees containing grammatical features. They convert the output of the LFG parser to dependency trees in order to have two parallel versions of their original data: (i) a gold standard Treebank, and (ii) a dependency Treebank by converting the parser output which contains additional grammatical features. They extend the gold standard Treebank with additional information from the corresponding LFG analyses. MaltPaser is then trained on the enhanced gold standard Treebank. Their results showed a small improvement in accuracy when applied to English and German.

A similar work in this area is conducted by Sagae and Miyao (2007). They constrain a Head-driven Phrase Structure Grammar (HPSG) parser with outputs from a data-driven parser. HPSG parsers use a small number of schemas for explaining general construction rules, and a large number of lexical entries for expressing word-specific syntactic and semantic constraints. HPSG parse trees are converted to Context Free Grammar style (CFG-style) trees. A dependency Treebank is then extracted from the CFG-style trees. The dependency Treebank is used for training a dependency data-driven parser, such as MaltPaser and MSTParser. Outputs from data-driven parsers are then used to constrain the HPSG parser. During HPSG parsing process, the lexical head of each partial parse tree is stored and in each schema application the head child is determined. Having such information about the head child and the lexical head, the dependency produced by the schema application is identified and whether the schema application violates the dependencies in the dependency Treebank is checked. The HPSG parser is forced to produce parse trees that are consistent with

the dependency trees. This approach is tested on English and some improvements in accuracy were achieved.

5. Parser hybridisation

In this section, we describe the steps for implementing a hybrid parser. We have implemented a data-driven dependency parser and we integrated a scoring technique into it in order to easily convert it to a hybrid parser. The parser is driven by data where a machine learning algorithm is used for parser training, but it is guided by a set of dependency relations that behaves as grammatical rules for constraining the parser to produce more accurate analysis. The first stage of our work was to obtain a dependency format treebank data because we are implementing a data-driven dependency parser. We then experimented with a number of machine learning algorithms in order to identify an algorithm that can classify our data accurately so that we extract question:answer pairs from its output and use them for guiding the parser. Finally, we have extracted a set of dependency relations from the dependency treebank so that we restrict the parser using these relations to produce correct analyses. The goal of using dependency relations is for testing the concept that we are using for hybridizing the data-driven parser and conduct preliminary experiments to examine how we can improve parser accuracy while keeping it reasonably efficient and robust.

5.1. From phrase structure to dependency structure

The Penn Arabic Treebank (PATB) (Maamouri and Bies, 2004) is a large collection of annotated modern standard Arabic text containing rich linguistic information, which is appropriate for parsing Arabic texts. However, the main challenge in using PATB is that it is based on phrase structure trees but the parser we are attempting to develop is a dependency based parser. We opt for dependency parsing because it is proven that dependency parsing can be robust, efficient and fairly accurate (Nivre, 2006; MacDonald, 2006). In order to make the PATB data appropriate for our use, we have converted it to a dependency format. The principle of the conversion from phrase structures to dependency structures is described clearly by (Xia, 2001) as (i) use the head percolation table for marking the head child of each node in a constituency format, and (ii) make the head of each non-head child depend on the head of the head-child in the dependency structure.

5.2 Dependency relations extraction from PATB

Once we have a dependency format of the PATB, we have extracted dependency relations between words from it. The technique that we have used for extracting dependency relations from a bracketed dependency tree is simple, which we describe below:

1. Get the head of the tree, which is the first item on the dependency tree.
2. Get the daughter(s) of the head, which is the next list of items in the dependency tree that follows the head.
3. Establishes dependency relationship between the head and the head of the daughter(s) and store the relationship in a set of the relations.
4. If there is more than one daughter for the head, then process each daughter in turn by repeating from step 1 to 3. otherwise, the daughter is not the head of anything and we can terminate.

[ate, [man, [the]], [apple, [an]]]
Fig. 1: Bracketed dependency tree

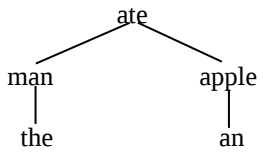

Fig. 2: Dependency tree

[[ate>man], [ate>apple], [man>the], [apple>an]]
Fig. 3: Dependency relation set

For example, we can extract the dependency relations from the bracketed tree in Fig. 1 using the technique we have described above. We first get the head of the tree, i.e., the first item in the list, which is “ate”. second, we get the daughters of “ate”, which is the lists ([man, [the]] and [apple, [an]]). We then establish dependency relations[1] between “ate” and its daughters, which would be “ate” as the head of “man” and “ate” as the head of “apple”, where “man” and “apple” are the head of the daughters of “ate”, i.e., “man” and “apple” are the first items in the lists [man, [the]] and [apple, [an]]. Thirdly, we store the relations in a set of relations as in Fig. 3. Since the head (ate) has more daughters, we process each of its daughter starting from the first daughter which is [man, [the]]. Starting from step 1 above, we choose “man” as the head and establish relations between “man” and its daughter(s) which is “the”. Since the daughter of “man” does not have any daughter of its own, we process the remaining daughters of “ate” until all of the daughters of “ate” and its sub-daughters are processed.

5.3 Developing a dependency data-driven parser

We have developed a parser based on shift-reduce algorithm (Aho and Ullman, 1972) with dynamic programming. The parser process an agenda, where an agenda contains a list of parse states. A parse state consists of data structures, a set of dependency relations, a score for each state, and a parse action. States on the agenda are sorted by their scores in descending order. The parser proceeds by processing the first state on the agenda. Processing each state results in creating new states, which are added to the agenda. The data structures consist of a queue of input strings, and a stack of tokens (which may include processed strings). Input strings

1 Relations between head and daughter are represented using '>' symbol.

contain various features such as: part-of-speech tags, word forms, word start position in the sentence, word end position in the sentence, and parse actions that may have been applied to them.

The parser performs three operations on the queues and the stacks: (i) shift, (ii) left-reduce, and (iii) right-reduce, where each of these operations results in a new parse state. Shift operation moves the first item from the queue to the top of the stack. Left-reduce operation makes the item from the stack a dependency parent of the item at the beginning of the queue. Right-reduce operation makes the first item on the queue a dependency parent of an item on the stack. Each parse operation creates new states which are stored in an agenda and the parser then process each state until it reaches a final parse state. If the parser reaches a state where there is an empty queue and a stack with one item, in which the item's start and end position on the stack covers the whole sentence length in question, then the parser successfully parsed the given set of input. Otherwise, the parse is unsuccessful.

In order to convert this parser to a hybrid parser where we uses the extracted dependency relations as grammatical rules for constraining the parser, we have implemented a scoring algorithm into the parser which assigns different scores to parse states. We can briefly describe our scoring technique below:

(a) W_{ML} is 1 if the parser follows machine learning suggestion, otherwise it is 0.

(b) W_G is 0 if the machine learning suggestion leads to a grammatical analysis, i.e., the suggested dependency relations by the machine learning algorithm conform to the relations in the set of dependency relations we have extracted from the PATB. Otherwise it is -0.5.

(c) A is the sum of $(ML * W_{ML}) + (G * W_G)$, where ML and G are weights that are used to determine the parser type. Given 0 weight to G then the parser follows machine learning suggestions and ignores Grammatical rules, which makes it a data-driven parser. Given 0 weight to ML then the parser follows grammatical rules and ignores the machine learning suggestions, which makes it a grammar-driven parser. Given intermediate weights to ML and G then the parser uses the machine learning suggestions and the grammatical rules, which makes it a hybrid parser.

Once we obtain scores for each parse states, we then sort all states in the agenda in descending order (because states with the highest score indicate that they are suggested by the machine learning algorithm and they also leads to correct analysis).

We have used a software toolkit, weka , which contains a large number of machine learning algorithms in order to experiment with a number of machine learning algorithms for parser training by supplying it with data containing different parse states. Our aim was to train the parser to perform appropriate parse actions in different situations. We have integrated a J48 algorithm in the parser due to its high classification accuracy (91%) compared with some other algorithms, such as Support Vector Machine (SVM) algorithm. We have used the output of the J48 algorithm for extracting question:answer pairs for guiding the parser to perform shift action or reduce action deterministically. Finally, we have conducted some initial experiments on the parser by running it as pure data-driven parser by given a weight of 0 to G and 1 ML. We have also experimented with the parser by running it as a hybrid parser to evaluate its accuracy.

6. Preliminary results

We have conducted some experiments on the parser by running it as a purely data-driven parser. The parser, by construction, is efficient and robust, because we provide it with shift, left-reduce or right-reduce action at each parse step, which deterministically lead to some analyses. However, the efficiency and robustness of the parser comes at the expense of its accuracy because the parser is guided by the machine learning alogrithm where it will always leads to some analyses, however, suggestions made by the machine learning algorithm are not always leading to correct analyses, hence the accuracy is affected. Having a hybrid parser that is driven by a machine learning algorithm but is constraint by rules, we can improve parser accuracy significantly. Table. 1 and Table. 2 show the differences between running the parser as purely data-driven and running as hybrid. We have trained the parser on 10000 words, and tested with 10000 words.

Training data (No. Sentences	Testing date (No. Sentences	Labelled attachment score	Unlabelled attachment score	Label Accuracy	Time Taken (seconds)
70000	7000	48.8%	48.8%	88.7%	142

Table. Data-driven parser testing

Training data (No. Sentences	Testing date (No. Sentences	Labelled attachment score	Unlabelled attachment score	Label Accuracy	Time Taken (seconds)
70000	7000	72.5	72.5	94.7	612

Table. Hybrid parser testing

7. Future work

We would like to find out how the hybrid parser performs when it is evaluated using grammatical rules that are linguistically sound. We have conducted these preliminary tests using dependency relations extracted from the PATB which behave as grammatical rules for restricting the parser to output analysis that obey a set of dependency relations. We anticipate that using linguistically sound grammatical rules may have an interesting impact on the hybrid parser, which we are planning to investigate it in the near future. In order to prepare grammatical rule for testing our hybrid parser, we are planning to produce a large number of grammatical rule using linguistic data available from the PATB and in-house grammar driven parser (Parasite).

8. Conclusion

Problems associated with using grammar-driven approaches and data-driven approaches are discussed in this paper. The main structural complexities of Arabic are identified and briefly described. We have very briefly highlighted the techniques that we have used for

converting the Penn Arabic Treebank form phrase structure to dependency format, and we also briefly highlighted a technique for extracting dependency relations from dependency treebanks. The first stage of our approach to hybrid parsing is explained, we also described our technique for developing a hybrid parser that directly combines features from data-driven approaches and grammar-driven approaches. We have presented our preliminary results for the parser the results at this stage is encouraging. The parser is tested on Arabic because it is a complex language, compare to some other languages, hence it provides a rigorous test-bed. Finally, various related works in hybrid parsing approaches for natural language processing is identified and briefly described.

References

Aho, A., V. & Ullman, J., D. (1972). *The Theory of Parsing, Translation, and Compiling*, Vol. 1, Prentic-Hall.

Attia, M. A. (2008). *Handling Arabic Morphological and Syntactic Ambiguities within the LFG Framework with a View to Machine Translation*, PhD Thesis, School of Languages, Linguistics and Cultures, Manchester University.

Baptista, M. (1995). On the Nature of Pro-drop in Capeverdean Creole, 5: 3–17.

Chomsky, N. (1981). *Lectures on Goverment and Binding*, Dordrecht: Foris.

Collins, M. (2003). Head-Driven Statistical Models for Natural Language Parsing, Comput. Linguist. 29(4): 589–637.

Crystal, D. (1980). *A First Dictionary of Linguistics and Phonetics*, Deutsch, London.

Daimi, K. (2001). Identifying Syntactic Ambiguities in Single-parse Arabic Sentence, 35: 333–349.

Farghaly, A. & Shaalan, K. (2009). Arabic Natural Language Processing: Challenges and Solutions, *ACM Computing Surveys* 8(4): 1–22.

Kaplan, R. M., Riezler, S., King, Tracy, H.,Maxwell III, John, T., Vasserman, A. & Crouch, R. (2004). Speed and accuracy in shallow and deep stochastic parsing, *Proceedings of Human Langauge Technology and the Conference of the North American Chapter of the Association for Computational Linguistics* (HLT-NAACL, pp. 97–104.

Lee, C., Day, M., Sung, C., Lee, Y., Jiang, T., Wu, C., Shih, C., Chen, Y. & Hsu, W. (2008). Boosting Chinese Question Answering with Two Lightweight Methods: ABSPs and SCO-QAT, 7(4): 12:1–12:29.

Maamouri, M. & Bies, A. (2004). Developing an Arabic treebank: Methods, guidelines, procedures, and tools, in A. Farghaly & K. Megerdoomian (eds), COLING 2004 *Computational Approaches to Arabic Script-based Languages*, COLING, Stroudsburg, PA, USA, pp. 2–9.

MacDonald, R. (2006). *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*, PhD Thesis, Computer and Information Science, the University of Pennsylvania.

Nelken, R. & Shieber, Stuart, M. (2005). Arabic Diacritization Using Weighted Finite-State Transducers, *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, Semitic '05*, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 79–86.

Nivre, J. (2006). Inductive Dependency Parsing, Springer.

Øvrelid, L., Kuhn, J. & Spreyer, K. (2009). Improving data-driven dependency parsing using large-scale lfg grammars, *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 37–40. URL: http://dl.acm.org/citation.cfm

Ramsay, A. & Mansour, H. (2006). Local Constraints on Arabic Word Order, *Proceedings of the 5th international conference on Advances in Natural Language Processing*, FinTAL'06, Springer-Verlag, Berlin, Heidelberg, pp. 447–457.

Sagae, K. & Miyao, Y. (2007). Hpsg parsing with shallow dependency constraints, I*n Proc*. ACL 2007.

Samuelsson, C. & Wir` n, M. (2000). *Parsing techniques*, Marcel Dekker.

Xia, F. & Palmer, M. (2001). Converting dependency structures to phrase structures, *Proceedings of the first international conference on Human language*
technology research, HLT ’01, Association for Computational Linguistics,

Stroudsburg, PA, USA, pp. 1–5.

URL: http://dx.doi.org/10.3115/1072133.1072147

Zitouni, I., Sorensen, Jaffery, S. & Sarikaya, R. (2006). Maximum Entropy Based
Restoration of Arabic Diacritics, *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the*

Association for Computational Linguistics, Association for Computational
Linguistics, Stroudsburg, PA, USA, pp. 577–584.